The Beer Drinkers Example

- Show all drinkers who like all the beers served somewhere for less than $3.00.

- Show all drinkers who like only the beers served somewhere for less than $3.00.

- Show all drinkers who like all the beers served exclusively for less than $3.00.

- Show all drinkers who like only the beers served exclusively for less than $3.00.

The Beer Drinkers Example

- Show all drinkers who frequent some bars that serve no beers they like.

- Show all drinkers who frequent bars that serve all the beers they like for less than $3.00.

The Beer Drinkers Example

- Show the cheapest beer.

- Show the bar that serves the cheapest beer.

- Show the drinkers who go to a bar even though all the beers they like at the bar are available cheaper elsewhere. (Show the bar, as well.)

The Beer Drinkers Example

- Show all drinkers that share a beer they like.

- Show all drinkers that share all the beers they like.

- Show the transitive closure of the first relationship.

Query "Safety"

The tables in databases are assumed to be finite. But it is trivial to come up with queries whose results are infinite. For example:

$$\exists br(\neg \mathbf{Likes}(\mathsf{hodas}, br))$$

Why is this infinite?

We need a testable restriction on queries that will guarantee that the results are finite. One property that was proposed is called *domain independence*. But it turns out that this is not a decidable property. The solution is *safety* which is a testable syntactic property that puts restrictions on the form of a query (and, in particular, how quantifiers and negations are used.)

A safe version of the above query might be:

$$\begin{array}{lll} \exists br(& \neg \mathbf{Likes}(\mathsf{hodas}, br) \wedge & \\ & (\ \exists dr(\mathbf{Likes}(dr, br)) \vee & \\ & \exists bar \exists pr(\mathbf{Serves}(bar, beer, price)) \ ) \ ) \end{array}$$